Linear Models

Previous Section's Recap

- Spatial data visualization
- Point pattern data: # of points and locations are random
- Parametric modeling of surface intensity

This Week's Key Concepts

- Linear Algebra
- Linear Model Overview
- Simulating Data in R
- Fitting Linear Models in R
- Bayesian Inference
- Fitting Bayesian Models with Stan

Matrices / Vectors

A matrix is an $n \times p$ object. Matrices are often denoted by a capital letter (or Greek symbol). A few common matrices will be

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} \end{pmatrix}$$

or

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \ddots & \sigma_n^2 \end{pmatrix}$$

Vectors are essentially one-dimension vectors and will be denoted with an underline. We will assume vectors are $q \times 1$ dimension unless noted with a transpose.

$$\underline{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

or

$$\underline{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{p-1} \end{pmatrix}$$

The transpose operator will be denoted by $\underline{y}^T = \begin{pmatrix} y_1 & y_2 & \cdots & y_n \end{pmatrix}$ or $\underline{y}'$, both of which would result in a $1 \times n$ vector.

Matrix Multiplication

The most important component in matrix multiplication is tracking dimensions.

Consider a simple case with

$$\underline{\hat{y}} = X \times \underline{\hat{\beta}},$$

where X is a 2×2 matrix, $\begin{pmatrix} 1 & 2 \\ 1 & -1 \end{pmatrix}$ and $\underline{\hat{\beta}} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$.

Then

$$\underline{\hat{y}} = \begin{bmatrix} 1 \times 3 + 2 \times 2 \\ 1 \times 3 + (-1) \times 2 \end{bmatrix} = \begin{bmatrix} 7 \\ -1 \end{bmatrix}$$

In R, we use **%*%** for matrix multiplication.

```
X <- matrix(c(1,2, 1 ,-1), nrow = 2, ncol = 2, byrow = T); X
```

```
     [,1] [,2]
[1,]    1    2
[2,]    1   -1
```

```
#X <- matrix(c(1, 1, 2 ,-1), nrow = 2, ncol = 2); X
beta_hat <- matrix(c(3,2),nrow =2, ncol = 1); beta_hat
```

```
     [,1]
[1,]    3
[2,]    2
```

```
y_hat <- X %*% beta_hat; y_hat
```

```
     [,1]
[1,]    7
[2,]    1
```

Linear Model Specification

Linear models provide the foundation for most statistical analyses:

- *regression models(matrix notation):* $y = X\beta + \epsilon$, *where* $\epsilon \sim N(0, \Sigma)$ *and* $\Sigma = \sigma^2 I$

One assumption, that is often violated in spatial statistics is that the errors are independently distributed.

Simulating Data in R

Simulating "fake" data will be a cornerstone of fitting models in this class.

```
set.seed(01112021)
# initialize parameters
num_obs <- 100
beta <- 1
sigma <- 2

# simulate data
x <- runif(num_obs, min = -10, max = 10)
y <- x * beta + rnorm(num_obs, mean = 0, sd = sigma)
```

```
# create figure
tibble(x=x, y=y) %>%
  ggplot(aes(x=x, y=y)) +
  geom_point() +
  theme_bw() +
  geom_smooth(formula = 'y~x', method = 'lm')
```

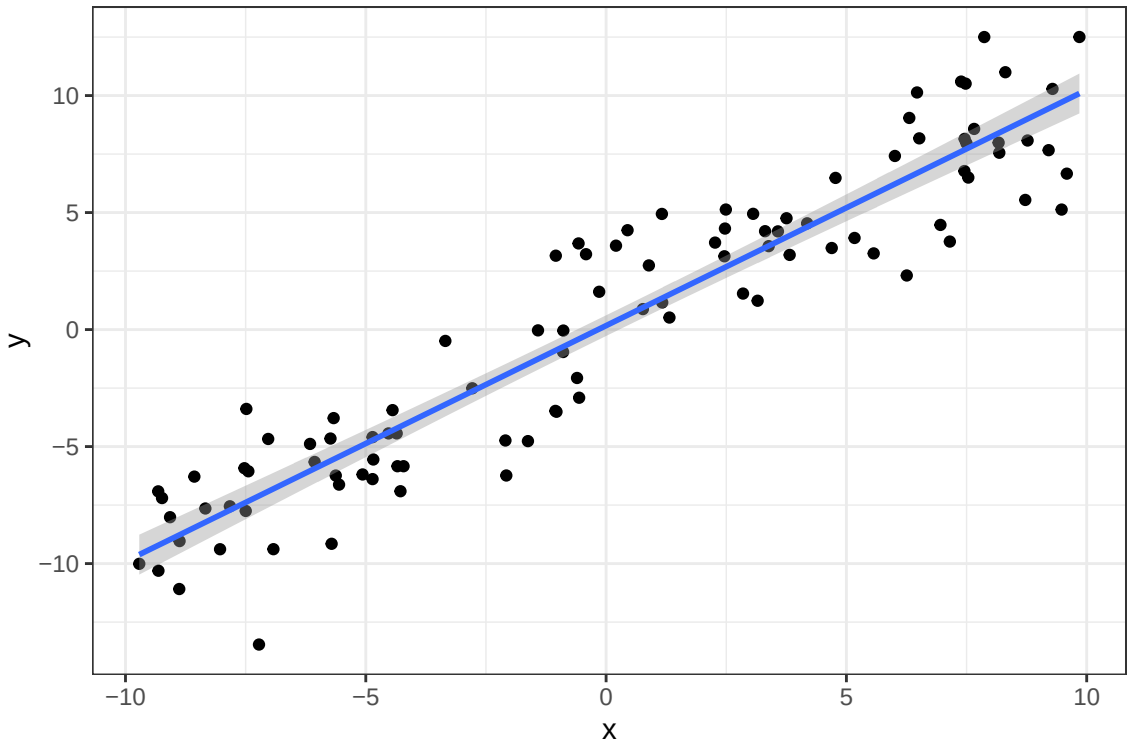


Figure 1: Scatterplot of synthetic data with best fit regression line.

Fitting Linear Models in R

The standard method for fitting linear models in R is with `lm()`.

```
fit_lm <- lm(y ~ x)
summary(fit_lm)
```

```
Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-6.3524 -1.4720  0.1297  1.5266  4.4066

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.16819    0.22278   0.755    0.452
x            1.00733    0.03755  26.827   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.228 on 98 degrees of freedom
Multiple R-squared:  0.8802,    Adjusted R-squared:  0.8789
F-statistic: 719.7 on 1 and 98 DF,  p-value: < 2.2e-16
```

A Bayesian alternative framework for fitting regression models is to use the **rstanarm** package and the associated `stan_glm()` functionality.

```
synthetic_data <- tibble(x = x, y = y)
fit_stan <- stan_glm(y ~ x, data = synthetic_data, refresh = 0)
print(fit_stan)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      y ~ x
 observations: 100
 predictors:   2
------
            Median MAD_SD
(Intercept) 0.2    0.2
```

```
x           1.0    0.0

Auxiliary parameter(s):
      Median MAD_SD
sigma 2.2    0.2

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Bayesian Inference

`stan_glm()` is a Bayesian procedure that uses Markov Chain Monte Carlo to generate samples that represent distribution for each parameter.

While there are coefficient estimates similar to `lm`

```
coef(fit_stan)
```

```
(Intercept)           x
  0.1747493   1.0075860
```

Rather than just a point estimate that maximizes the likelihood, the object actually contains a set of samples for each of the parameter values.

```
fit_stan %>% as.data.frame() %>% head(10)
```

```
   (Intercept)         x    sigma
1   0.18706157 1.0098023 2.529174
2   0.09967871 0.9939963 2.079094
3   0.08007354 0.9818507 2.215735
4   0.04590595 0.9796351 2.108727
5   0.02927909 0.9952074 2.133086
6   0.49101367 0.9909962 2.364030
7   0.26350585 0.9921205 2.271885
8   0.22166709 0.9987937 2.009585
9   0.10027736 1.0141302 2.488189
10  0.67024064 0.9608660 2.247425
```

While the estimation procedure and interval interpretations are fundamentally different, the actual values on the interval are fairly similar.

```
posterior_interval(fit_stan, prob = .95)
```

```
                 2.5%     97.5%
(Intercept) -0.2773379 0.6195185
x            0.9331275 1.0801073
sigma        1.9687029 2.5849950
```

```
confint(fit_lm)
```

```
                2.5 %    97.5 %
(Intercept) -0.2739015 0.6102871
x            0.9328161 1.0818436
```

So what is hiding in this model specification? **priors, of course**

```
synthetic_data <- tibble(x = x, y = y)
fit_stan <- stan_glm(y ~ x, data = synthetic_data, refresh = 0)
```

```
prior_summary(fit_stan)
```

```
Priors for model 'fit_stan'
------
Intercept (after predictors centered)
  Specified prior:
    ~ normal(location = 0.24, scale = 2.5)
  Adjusted prior:
    ~ normal(location = 0.24, scale = 16)

Coefficients
  Specified prior:
    ~ normal(location = 0, scale = 2.5)
  Adjusted prior:
    ~ normal(location = 0, scale = 2.7)

Auxiliary (sigma)
  Specified prior:
    ~ exponential(rate = 1)
  Adjusted prior:
    ~ exponential(rate = 0.16)
------
See help('prior_summary.stanreg') for more details
```

With Bayesian inference, Markov Chain Monte Carlo (MCMC) is used to compute the posterior distribution. The `stan_glm()` functions call STAN to implement MCMC behind the curtain to find posterior distributions of parameters. MCMC can be written in any language, but STAN has some nice advantages (Hamiltonian Monte Carlo).

Motivating Dataset: Washington (DC) housing dataset

Hopefully the connections to statistics are clear, using X and β, but let's consider a motivating dataset.

This dataset contains housing information from Washington, D.C. It was used for a STAT532 exam, so apologize in advance for any scar tissue.

```
DC <- read_csv('https://math.montana.edu/ahoegh/teaching/stat532/data/DC.csv')
```

```
Rows: 2303 Columns: 12
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (6): AC, CNDTN, FULLADDRESS, ASSESSMENT_NBHD, WARD, QUADRANT
dbl (6): BATHRM, HF_BATHRM, BEDRM, STORIES, PRICE, LANDAREA

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
DC %>% group_by(WARD) %>%
  summarize(`Average Price (millions of dollars)` = mean(PRICE)/1000000, .groups = 'drop') %>
  kable(digits = 3)
```

WARD	Average Price (millions of dollars)
Ward 1	0.879
Ward 2	1.919
Ward 3	1.294
Ward 4	0.693
Ward 5	0.592
Ward 6	0.856
Ward 7	0.321
Ward 8	0.306

```
DC %>% group_by(BEDRM) %>%
  summarize(`Average Price (millions of dollars)` = mean(PRICE)/1000000, .groups = 'drop') %>
  kable(digits = 3)
```

BEDRM	Average Price (millions of dollars)
0	0.195
1	0.442
2	0.479
3	0.620
4	0.832
5	1.355
6	1.849
7	1.666
9	7.365

Regression Model

There are many factors in this dataset that can are useful to predict housing prices.

$$y_i = \beta_0 + \beta_1 * x_{SQFT,i} + \beta_2 x_{BEDRM,i} + \epsilon_i, \tag{1}$$

where y_i is the sales price of the i^{th} house, $x_{SQFT,i}$ is the living square footage of the i^{th} house, and $x_{BEDRM,i}$ is the number of bedrooms for the i^{th} house. Note this implies that we are treating bedrooms as continuous variables as opposed to categorical.

we usually write $\epsilon_i \sim N(0, \sigma^2)$. More on that soon.

In R we often write something like: `price ~ LANDUSE + BEDRM`.

Now let's write this model in matrix notation:

$$\underline{y} = X\underline{\beta} + \underline{\epsilon}, \tag{2}$$

where $\underline{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$, $X = \begin{bmatrix} 1 & x_{SQFT,1} & x_{BEDRM,1} \\ 1 & x_{SQFT,2} & x_{BEDRM,2} \\ \vdots & \vdots & \vdots \\ 1 & x_{SQFT,n} & x_{BEDRM,n} \end{bmatrix}$,$\underline{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$, and $\underline{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$

Now what are the implications of:

$\epsilon_i \sim N(0, \sigma^2)$ or $\underline{\epsilon} \sim N(\underline{0}, \Sigma)$, where

$$\Sigma = \sigma \times \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

These are equivalent statements and both imply that y_i and y_j are conditionally independent given X. In other words, after controlling for predictors (ward, square footage), then the price of house i gives us no additional information about price of house j.

Diagonal Matrices

The matrix we previously specified, is referred to as a diagonal matrix. This is often denoted with

$$I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix},$$

where n is the dimension. Note this is also just shortened to I.

Correlation Matrices

It turns out that I is the special case of what is referred to as a correlation matrix.

A correlation matrix is:

- is symmetric
- contains ones on the diagonal
- contains correlation terms on the off diagonal
- is positive definite (more later)

Similarly Σ is often referred to as a variance - covariance matrix (or just a covariance matrix). A covariance matrix:

- is symmetric
- contains variance terms on the diagonal
- contains covariance terms on the off diagonal
- is positive definite (more later)

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \ddots & \sigma_n^2 \end{pmatrix}$$

Multivariate Normal Distribution

Formally, our matrix notation has used a multivariate normal distribution.

$$\underline{y} = X\underline{\beta} + \underline{\epsilon}, \tag{3}$$

where $\underline{\epsilon} \sim N(\underline{0}, \Sigma)$, which also implies $\underline{y} \sim N(X\underline{\beta}, \Sigma)$.

Partitioned Matrices

Now consider splitting the sampling units into two partitions such that $\underline{y} = \begin{bmatrix} \underline{y}_1 \\ \underline{y}_2 \end{bmatrix}$. Then,

$$\begin{bmatrix} \underline{y}_1 \\ \underline{y}_2 \end{bmatrix} \sim N\left(\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \underline{\beta}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{22} \end{bmatrix} \right)$$

Fundamentally, there is no change to the model, we have just created "groups" by partitioning the model. Do note that Σ_{11} is an $n_1 \times n_1$ covariance matrix.

$$\Sigma_{11} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n_1} \\ \sigma_{22} & \sigma_2^2 & \cdots & \sigma_{2n_1} \\ \sigma_{31} & \sigma_{32} & \ddots & \sigma_{3n_1} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n_1 1} & \sigma_{n_1 2} & \ddots & \sigma_{n_1}^2 \end{bmatrix}$$

However, while $\Sigma_{12} = \Sigma_{21}^T$, neither of these are necessarily symmetric matrices. They also do not have any variance components, but rather just covariance terms. Σ_{12} will be an $n_1 \times n_2$ matrix.

$$\Sigma_{11} = \begin{bmatrix} \sigma_{1,n_1+1} & \sigma_{1,n_1+2} & \cdots & \sigma_{1,n_1+n_2} \\ \sigma_{2,n_1+1} & \sigma_{2,n1+2} & \cdots & \sigma_{2,n_1+n_2} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n_1,n_1+1} & \sigma_{n_1,n_1+2} & \ddots & \sigma_{n_1,n_1+n_2} \end{bmatrix}$$

Conditional Multivariate Normal

Here is where the magic happens with correlated data. Let $\underline{y_1}|\underline{Y_2} = \underline{y_2}$ be a conditional distribution for $\underline{y_1}$ given that $\underline{y_2}$ is known. Then

$$\underline{y_1}|\underline{y_2} \sim N\left(X_1\beta + \Sigma_{12}\Sigma_{22}^{-1}\left(\underline{y_2} - X_2\beta\right), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right)$$

Now let's consider a few special cases (in the context of the DC housing dataset.)

1. Let $\Sigma = \sigma^2 I$, then the batch of houses in group 1 are conditionally dependent from the houses in group 2 and

$$\underline{y_1}|\underline{y_2} \sim N\left(X_1\beta, \Sigma_{11}\right)$$

2. Otherwise, let $\Sigma = \sigma^2 H$ and we'll assume Σ_{12} has some non-zero elements. Then we have a more precise estimate of $\underline{y_1}$ as $\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$ will be "less than" Σ_{11} (that positive definite thing). Furthermore, the mean will shift such that highly correlated observations such as houses in close proximity (local model structure) will tend to differ from the global mean in the same fashion.

First a quick interlude about matrix inversion. The inverse of a symmetric matrix is defined such that $E \times E^{-1} = I$. We can calculate the inverse of a matrix for a 1×1 matrix, perhaps as 2×2, matrix and maybe even a 3×3 matrix. However, beyond that it is quite challenging and time consuming. Furthermore, it is also (relatively) time intensive for your computer.

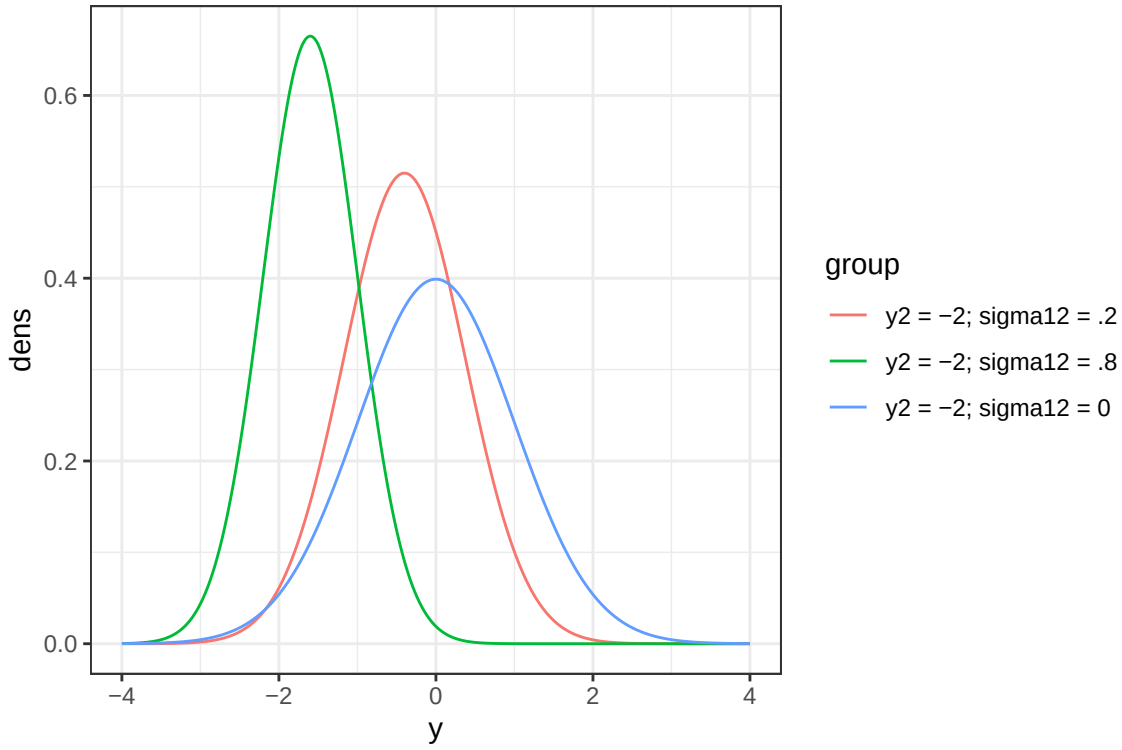
3. Let $n_1 = 1$ and $n_2 = 1$, then

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix} \right)$$

and

$$y_1|y_2 \sim N\left(\mu_1 + \sigma_{12}(\sigma_2^2)^{-1}\left(y_2 - \mu_2\right), \sigma_1^2 - \sigma_{12}(\sigma_2^2)^{-1}\sigma_{21}\right)$$

Now consider an illustration for a couple simple scenarios. Let $\mu_1 = \mu_2 = 0$ and $\sigma_1^2 = \sigma_2^2 = 1$. Now assume $y_2 = -2$ and we compare the conditional distribution for a few values of σ_{12}.



One last note, the marginal distributions for any partition $\underline{y_1}$ are quite simple.

$$\underline{y_1} \sim N\left(X_1\beta, \Sigma_{11}\right)$$

or just

$$y_1 \sim N\left(X_1\beta, \sigma_1^2\right)$$

if y_1 is scalar.